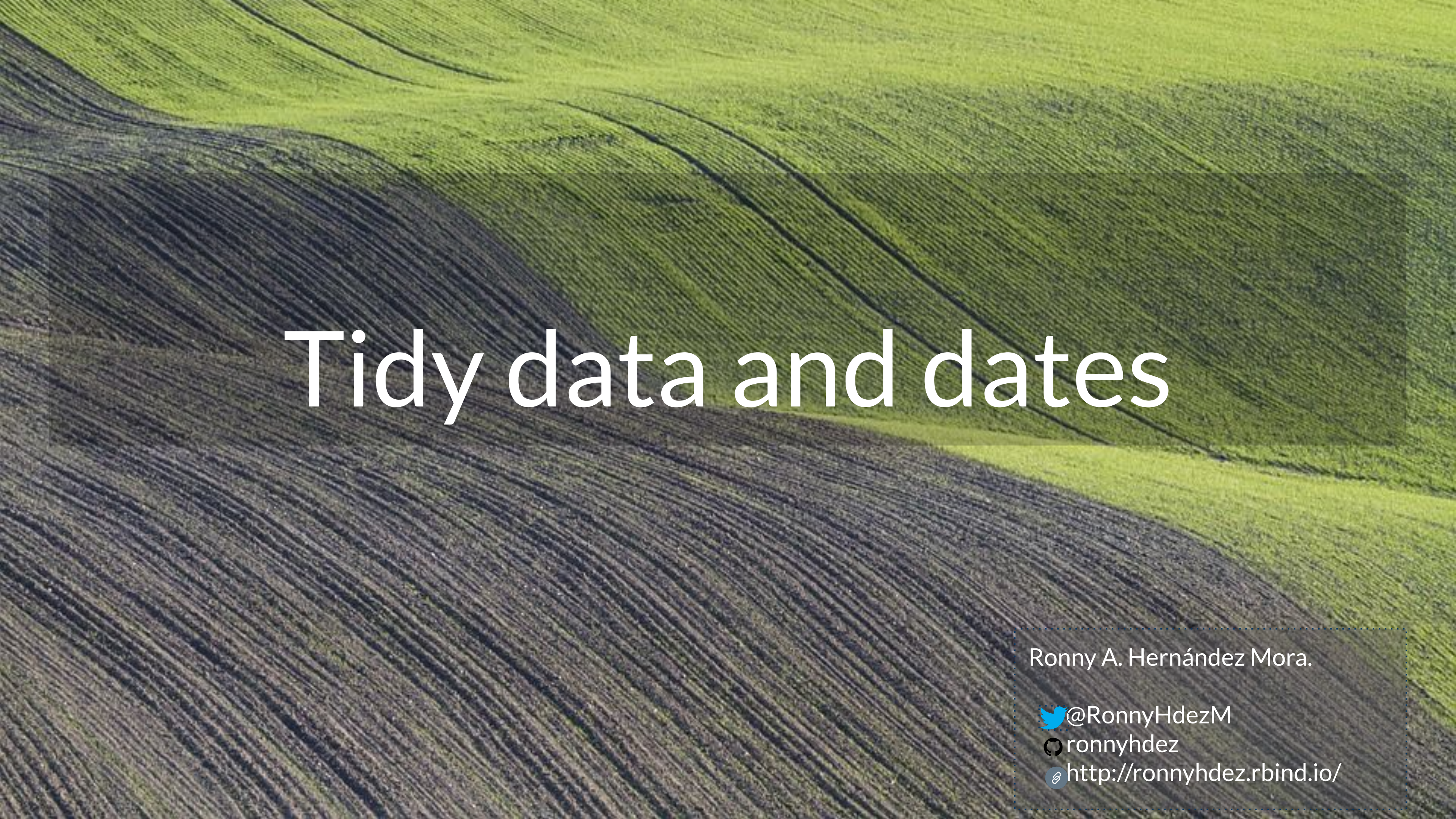
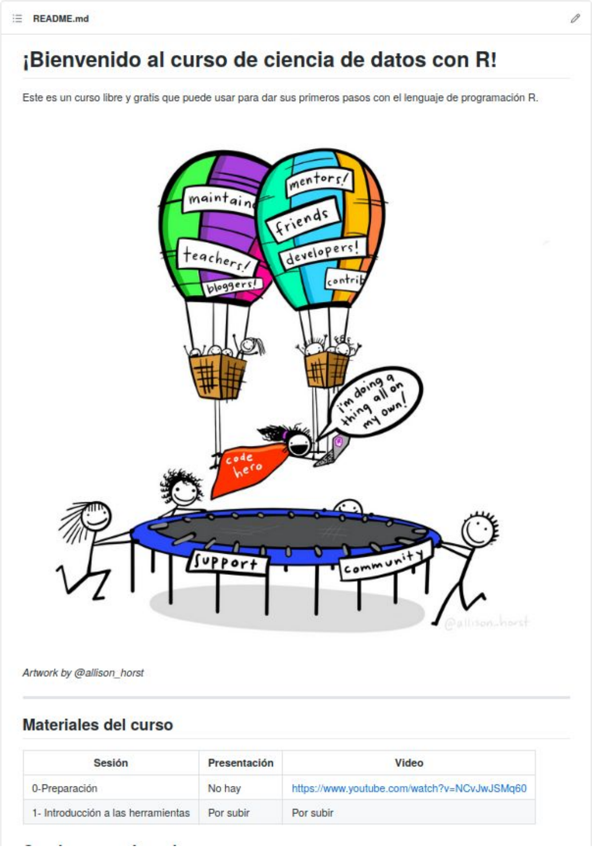


Tidy data and dates

Ronny A. Hernández Mora.

@RonnyHdezM
ronnyhdez
http://ronnyhdez.rbind.io/

Workshop materials



README.md

¡Bienvenido al curso de ciencia de datos con R!

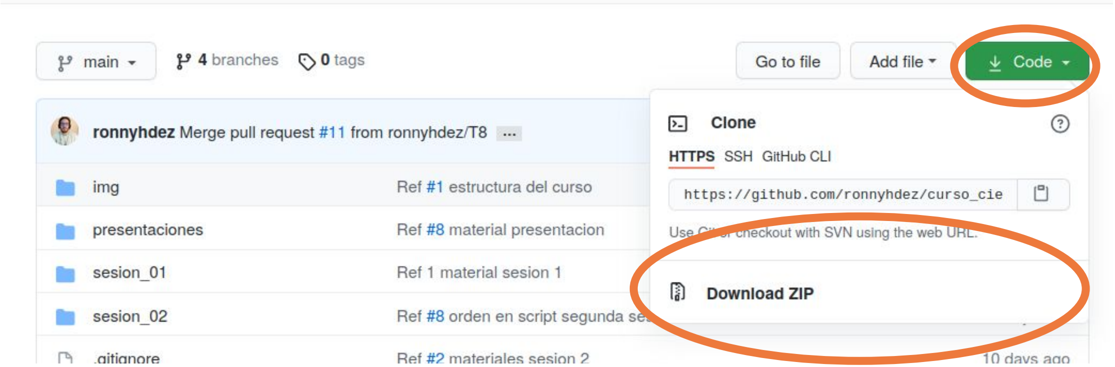
Este es un curso libre y gratis que puede usar para dar sus primeros pasos con el lenguaje de programación R.

Artwork by @allison_horst

Materiales del curso

Sesión	Presentación	Video
0-Preparación	No hay	https://www.youtube.com/watch?v=NCvJwJSMq60
1- Introducción a las herramientas	Por subir	Por subir

https://github.com/ronnyhdez/curso_ciencia_datos_r



ronnyhdez Merge pull request #11 from ronnyhdez/T8 …

img	Ref #1 estructura del curso
presentaciones	Ref #8 material presentacion
sesion_01	Ref 1 material sesion 1
sesion_02	Ref #8 orden en script segunda se
.gitignore	Ref #2 materiales sesion 2

10 days ago

Clone

HTTPS SSH GitHub CLI

```
https://github.com/ronnyhdez/curso_cie
```

Use Git or checkout with SVN using the web URL.

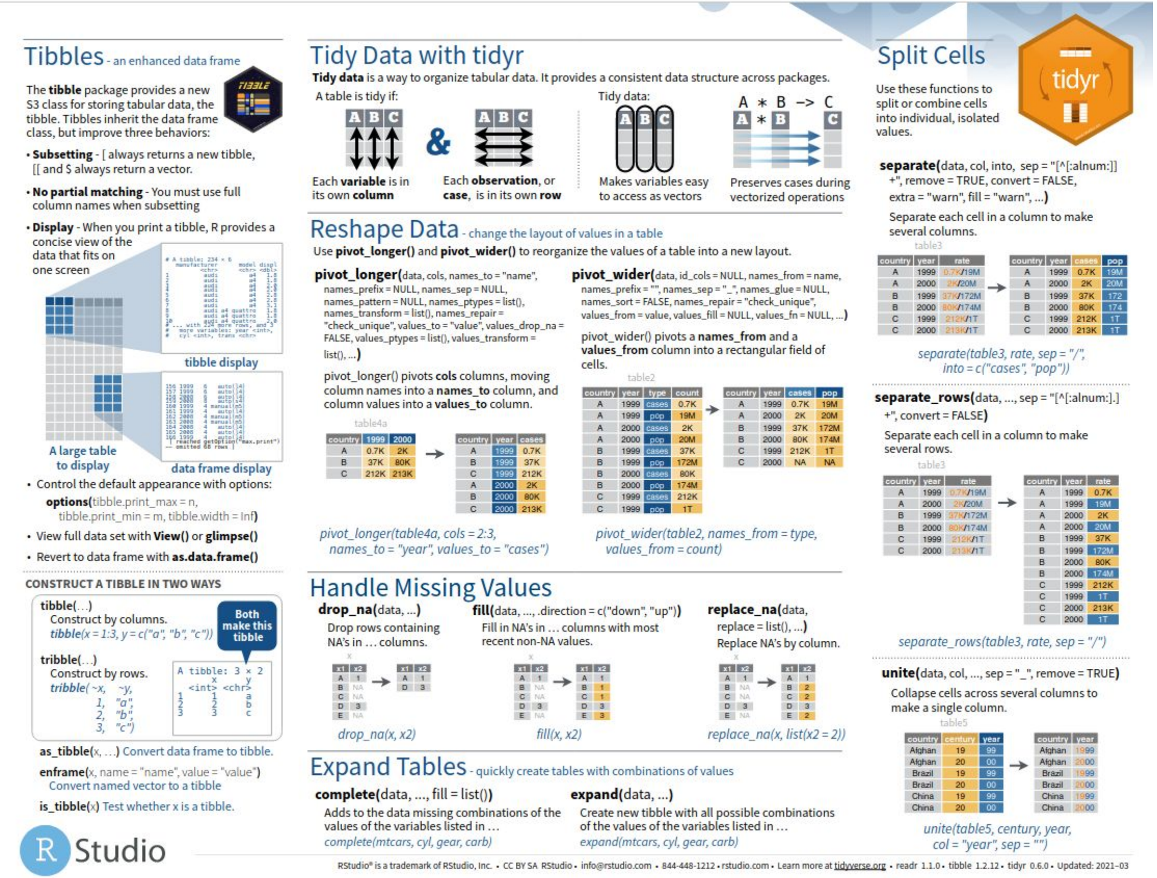
Download ZIP

What do we want from today’s session?

- Understand what is tidy data
- How to use tidyr
- Understand dates

Tidy data

RStudio cheatsheet



TIBBLE
A large table to display
tibble display
data frame display
A * B -> C
A B C
&
tidyr

Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- **Subsetting** - [always returns a new tibble, [[and $ always return a vector.
- **No partial matching** - You must use full column names when subsetting
- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen

```
# A tibble: 234 × 6
   manufacturer      model displ
          <chr>      <chr> <dbl>
1          audi         a4   1.8
2          audi         a4   1.8
3          audi         a4   2.0
4          audi         a4   2.0
5          audi         a4   2.8
6          audi         a4   2.8
7          audi         a4   3.1
8          audi a4 quattro   1.8
9          audi a4 quattro   1.8
10         audi a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

tibble display

```
156 1999  6   auto(l4)
157 1999  6   auto(l4)
158 2008  6   auto(l4)
159 2008  8   auto(s4)
160 1999  4 manual(m5)
161 1999  4   auto(l4)
162 2008  4 manual(m5)
163 2008  4 manual(m5)
164 2008  4   auto(l4)
165 2008  4   auto(l4)
166 1999  4   auto(l4)
 [ reached getOption("max.print")
 -- omitted 68 rows ]
```

A large table to display

data frame display

- Control the default appearance with options:
  options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)**
- View full data set with **View()** or **glimpse()**
- Revert to data frame with **as.data.frame()**

CONSTRUCT A TIBBLE IN TWO WAYS

tibble(…**)**
Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

tribble(…**)**
Construct by rows.

```
tribble( ~x,  ~y,
          1, "a",
          2, "b",
          3, "c")
```

Both make this tibble

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …**)** Convert data frame to tibble.

enframe(x, name = "name", value = "value"**)**
Convert named vector to a tibble

is_tibble(x**)** Test whether x is a tibble.

RStudio

Tidy Data with tidyr

Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.

A table is tidy if:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

Tidy data:

Makes variables easy to access as vectors

Preserves cases during vectorized operations

Reshape Data - change the layout of values in a table

Use **pivot_longer()** and **pivot_wider()** to reorganize the values of a table into a new layout.

pivot_longer(data, cols, names_to = "name", names_prefix = NULL, names_sep = NULL, names_pattern = NULL, names_ptypes = list(), names_transform = list(), names_repair = "check_unique", values_to = "value", values_drop_na = FALSE, values_ptypes = list(), values_transform = list(), ...**)**

pivot_longer() pivots **cols** columns, moving column names into a **names_to** column, and column values into a **values_to** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")

pivot_wider(data, id_cols = NULL, names_from = name, names_prefix = "", names_sep = "_", names_glue = NULL, names_sort = FALSE, names_repair = "check_unique", values_from = value, values_fill = NULL, values_fn = NULL, ...**)**

pivot_wider() pivots a **names_from** and a **values_from** column into a rectangular field of cells.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	NA	NA

pivot_wider(table2, names_from = type, values_from = count)

Handle Missing Values

drop_na(data, ...**)**
Drop rows containing NA's in … columns.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(x, x2)

fill(data, ..., .direction = c("down", "up")**)**
Fill in NA's in … columns with most recent non-NA values.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(x, x2)

replace_na(data, replace = list(), ...**)**
Replace NA's by column.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(x, list(x2 = 2))

Expand Tables - quickly create tables with combinations of values

complete(data, ..., fill = list()**)**
Adds to the data missing combinations of the values of the variables listed in …
complete(mtcars, cyl, gear, carb)

expand(data, ...**)**
Create new tibble with all possible combinations of the values of the variables listed in …
expand(mtcars, cyl, gear, carb)

Split Cells

Use these functions to split or combine cells into individual, isolated values.

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)**

Separate each cell in a column to make several columns.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

separate(table3, rate, sep = "/", into = c("cases", "pop"))

separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE**)**

Separate each cell in a column to make several rows.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

separate_rows(table3, rate, sep = "/")

unite(data, col, ..., sep = "_", remove = TRUE**)**

Collapse cells across several columns to make a single column.

table5

country	century	year
Afghan	19	99
Afghan	20	00
Brazil	19	99
Brazil	20	00
China	19	99
China	20	00

→

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

unite(table5, century, year, col = "year", sep = "")

https://www.rstudio.com/resources/cheatsheets/

What is tidy data?

course	**date**	**grade**	**estudents**
matematica	2020-05-28	excelente	34
historia del arte	2020-06-04	regular	20
computacion	2020-06-12	bueno	28

Each row is an **observation**

Each **column** is a **variable**

- Each **variable** must have its own **column**.
- Each **observation** must have its own **row**.
- Each **value** must have its own **cell**.

Is this tidy data?

course	date	grade	students/answers
matematica	2020-05-28	excelente	34/20
historia del arte	2020-06-04	regular	20/18
computacion	2020-06-12	bueno	28/12

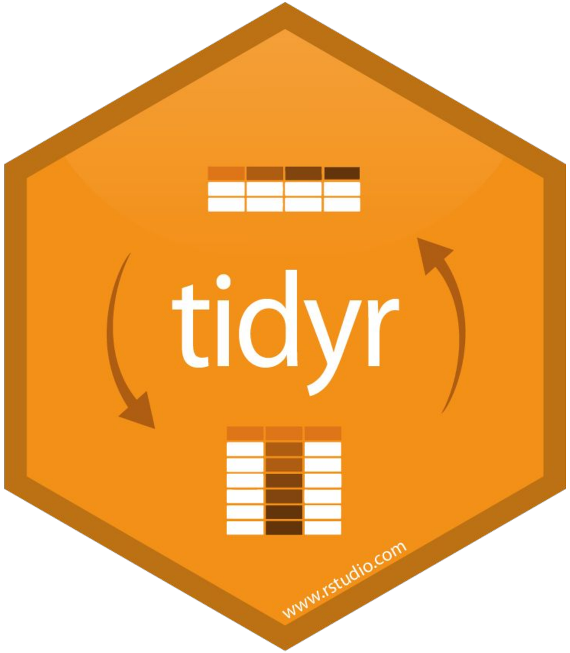
Is this tidy data?

course	date	grade	type	total
matematica	2020-05-28	excelente	estudiantes	34
matematica	2020-05-28	excelente	respuestas	20
historia del arte	2020-06-04	regular	estudiantes	20
historia del arte	2020-06-04	regular	respuestas	18
computacion	2020-06-12	bueno	estudiantes	28
computacion	2020-06-12	bueno	respuestas	12

How to make my data tidy?

```
pivot_longer( arguments )
```

```
pivot_wider( arguments )
```



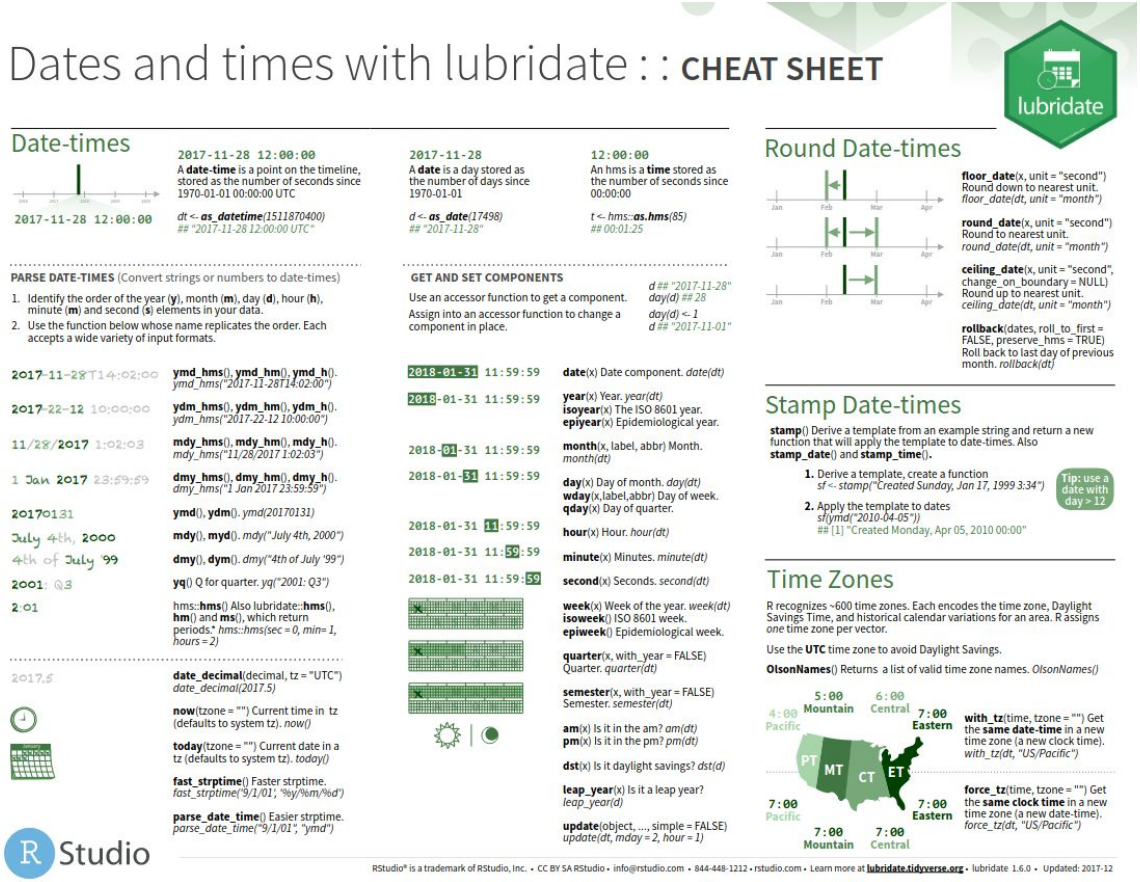
https://tidyr.tidyverse.org

https://www.garrickadenbuie.com/project/tidyexplain/

Break 12:00 a 12:10

Dates with lubridate

RStudio cheat sheets



Dates and times with lubridate : : CHEAT SHEET

lubridate

Date-times

2017-11-28 12:00:00

2017-11-28 12:00:00
A **date-time** is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC

*dt <- **as_datetime**(1511870400)*
## "2017-11-28 12:00:00 UTC"

2017-11-28
A **date** is a day stored as the number of days since 1970-01-01

*d <- **as_date**(17498)*
## "2017-11-28"

12:00:00
An hms is a **time** stored as the number of seconds since 00:00:00

*t <- hms::**as.hms**(85)*
## 00:01:25

PARSE DATE-TIMES (Convert strings or numbers to date-times)

1. Identify the order of the year (**y**), month (**m**), day (**d**), hour (**h**), minute (**m**) and second (**s**) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00 — **ymd_hms**(), **ymd_hm**(), **ymd_h**(). *ymd_hms("2017-11-28T14:02:00")*

2017-22-12 10:00:00 — **ydm_hms**(), **ydm_hm**(), **ydm_h**(). *ydm_hms("2017-22-12 10:00:00")*

11/28/2017 1:02:03 — **mdy_hms**(), **mdy_hm**(), **mdy_h**(). *mdy_hms("11/28/2017 1:02:03")*

1 Jan 2017 23:59:59 — **dmy_hms**(), **dmy_hm**(), **dmy_h**(). *dmy_hms("1 Jan 2017 23:59:59")*

20170131 — **ymd**(), **ydm**(). *ymd(20170131)*

July 4th, 2000 — **mdy**(), **myd**(). *mdy("July 4th, 2000")*

4th of July '99 — **dmy**(), **dym**(). *dmy("4th of July '99")*

2001: Q3 — **yq**() Q for quarter. *yq("2001: Q3")*

2:01 — hms::**hms**() Also lubridate::**hms**(), **hm**() and **ms**(), which return periods.* *hms::hms(sec = 0, min= 1, hours = 2)*

2017.5 — **date_decimal**(decimal, tz = "UTC") *date_decimal(2017.5)*

now(tzone = "") Current time in tz (defaults to system tz). *now()*

today(tzone = "") Current date in a tz (defaults to system tz). *today()*

fast_strptime() Faster strptime. *fast_strptime('9/1/01', '%y/%m/%d')*

parse_date_time() Easier strptime. *parse_date_time("9/1/01", "ymd")*

GET AND SET COMPONENTS

Use an accessor function to get a component.

d ## "2017-11-28"
day(d) ## 28

Assign into an accessor function to change a component in place.

day(d) <- 1
d ## "2017-11-01"

2018-01-31 11:59:59 — **date**(x) Date component. *date(dt)*

2018-01-31 11:59:59 — **year**(x) Year. *year(dt)*
isoyear(x) The ISO 8601 year.
epiyear(x) Epidemiological year.

2018-01-31 11:59:59 — **month**(x, label, abbr) Month. *month(dt)*

2018-01-31 11:59:59 — **day**(x) Day of month. *day(dt)*
wday(x,label,abbr) Day of week.
qday(x) Day of quarter.

2018-01-31 11:59:59 — **hour**(x) Hour. *hour(dt)*

2018-01-31 11:59:59 — **minute**(x) Minutes. *minute(dt)*

2018-01-31 11:59:59 — **second**(x) Seconds. *second(dt)*

week(x) Week of the year. *week(dt)*
isoweek() ISO 8601 week.
epiweek() Epidemiological week.

quarter(x, with_year = FALSE) Quarter. *quarter(dt)*

semester(x, with_year = FALSE) Semester. *semester(dt)*

am(x) Is it in the am? *am(dt)*
pm(x) Is it in the pm? *pm(dt)*

dst(x) Is it daylight savings? *dst(d)*

leap_year(x) Is it a leap year? *leap_year(d)*

update(object, ..., simple = FALSE) *update(dt, mday = 2, hour = 1)*

Round Date-times

floor_date(x, unit = "second") Round down to nearest unit. *floor_date(dt, unit = "month")*

round_date(x, unit = "second") Round to nearest unit. *round_date(dt, unit = "month")*

ceiling_date(x, unit = "second", change_on_boundary = NULL) Round up to nearest unit. *ceiling_date(dt, unit = "month")*

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) Roll back to last day of previous month. *rollback(dt)*

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also **stamp_date**() and **stamp_time**().

1. Derive a template, create a function
sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Apply the template to dates
sf(ymd("2010-04-05"))
## [1] "Created Monday, Apr 05, 2010 00:00"

Tip: use a date with day > 12

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns *one* time zone per vector.

Use the **UTC** time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. *OlsonNames()*

with_tz(time, tzone = "") Get the **same date-time** in a new time zone (a new clock time). *with_tz(dt, "US/Pacific")*

force_tz(time, tzone = "") Get the **same clock time** in a new time zone (a new date-time). *force_tz(dt, "US/Pacific")*

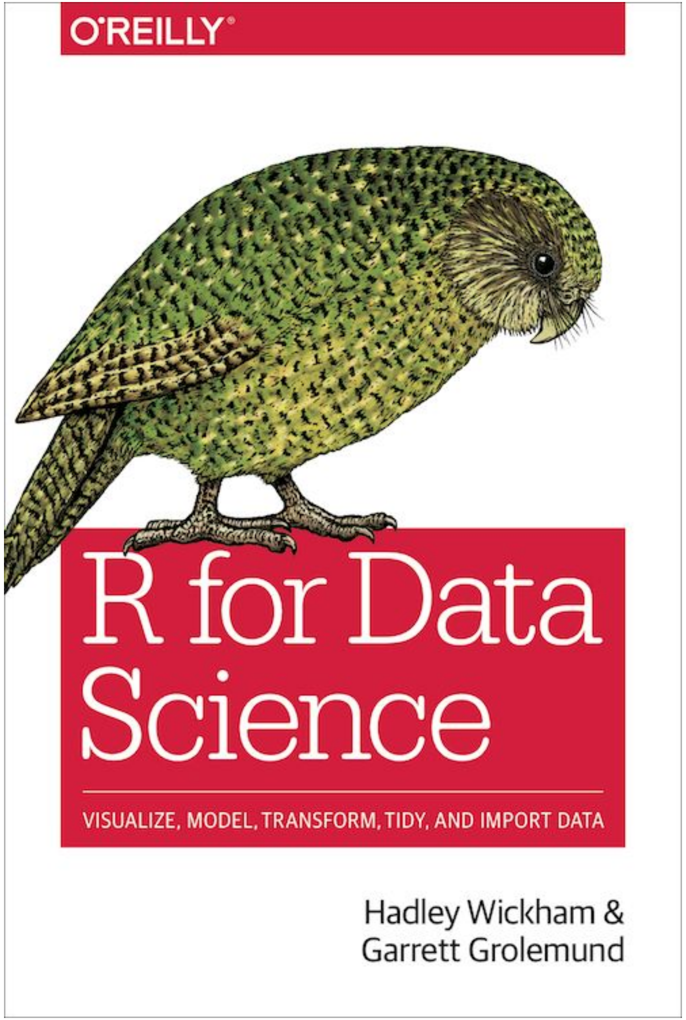
RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at lubridate.tidyverse.org • lubridate 1.6.0 • Updated: 2017-12

https://www.rstudio.com/resources/cheatsheets/

Advanced R
Wickham
Zuur • Ieno • Walker
Saveliev • Smith
Mixed Effects Models and Extensions in Ecology with R
The Pragmatic Programmer
20th Anniversary Edition
THINK LIKE A PROGRAMMER
SPRAUL
R for Data Science
Wickham & Grolemund
O'REILLY
Mastering Ubuntu Server
Third Edition
Jay LaCroix
Fundamentals of Data Visualization
Wilke
O'RE
LINUX BASICS FOR HACKERS
OCCUPYTHEWEB
FIFTH EDITION
Learning Python
Programación en C++
Algoritmos, estructuras de datos y objetos
L. Joyanes Aguilar
Raspberry Pi 3
Model B+
E1385

Recursos



O'REILLY®

R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

Hadley Wickham &
Garrett Grolemund

https://es.r4ds.hadley.nz/

¡Gracias!

Ronny A. Hernández Mora.

@RonnyHdezM
ronnyhdez
http://ronnyhdez.rbind.io/

ronny.hernandezm@gmail.com